REMARKS

This is in response to the Office Action of November 14, 2003. Claims 1-18 were rejected. Claims 3, 10, and 17 were amended. Claims 1-18 are pending.

Claims 3, 10, and 17 were rejected under 35 U.S.C. 112. Applicant has corrected a typographical error and has amended claim 17 to be dependent from claim 16. Claims 3 and 10 were amended in accordance with the Examiner's comments. Support for this amendment is found, for example, in paragraph 1146, which describes an embodiment having a UT field to indicate a type of user trace data record.

Claims 1, 2, 4, 6-9, 11, 13-15, and 18 were rejected under 35 U.S.C. 102(e) as being anticipated by Ayers et al. (U.S. Pat. No. 6,353,924). Claims 3 and 10 were rejected under 35 U.S.C. 103 as being obvious over Ayers in view of Baird (U.S. Pat. No. 5,848,264). Claims 5 and 12 were rejected under 35 U.S.C. 103(a) as being obvious over Ayers in view of the article "How Debuggers Work: Algorithms, Data Structures, and Architecture." Claims 16 and 17 were rejected under 35 U.S.C. 103(a) as being obvious over Ayers. Applicant respectfully traverses the rejections.

Applicant's claimed invention of independent claim 1 includes limitations corresponding to writing user trace data when a user trace data command is executed within a plurality of instructions. The user trace data is written to a user trace data register. A trace record that includes at least part of the user trace data is generated in response to detecting the write to the user trace data register. Independent claims 8, 15, 16, and 18 include similar limitations.

One benefit of Applicant's claimed invention is that it permits user control of trace records. In particular, embodiments of Applicant's invention permit a user to trace out any data value that can be written into a user trace data register by software, as described in paragraph 1148 of Applicant's specification. Trace data may be, for example, any general processor register value, any program variable, or any other debug-related information that is observable or computable during program execution, as described in paragraph 1148 of Applicant's specification. This improves, for example, the flexibility in tracing intermediate values, a small subset of data values, or specific points or aspects of program execution.

Ayers is directed towards the different problem of creating a crash instruction trace that includes all of the instructions that crashed and some of the instructions that preceded it. See, e.g., Ayers at column 2, lines 28-31. In Ayers, a block is a sequence of consecutive program instructions in which flow of control enters only at the beginning and leaves only at the end without halt or possibility of branching, except at the end. See, e.g., Ayers at column 5, lines 46-54.

Ayers teaches that instrumentation code is inserted into each block of a computer program. See, e.g., Ayers at column 5, lines 47-66. As each block executes, the instrumentation code writes out an identifier, which is written into a memory. See, e.g., Ayers at column 6, lines 3-10. Consequently a memory record that stores the block sequence of execution is created, as illustrated in Figure 3 of Ayers.

Note that the block sequence of execution is not traced out in Ayers. As illustrated in Figures 4 and 5 of Ayers, an event trigger 217 (FIG. 5) or 317 (FIG. 4) results in post-processing of the block sequence of execution. Ayers teaches that post-processor 217 (FIG. 5), 317 (FIG. 4) is triggered when it detects an event, such a crash or a user-defined event. See, e.g., Ayers at column 7, lines 9-14. When the post processor is triggered it writes out the sequence record of execution of the blocks and replaces each entry in a block with the set of program counters for the block in order to produce a trace record of program counter values. See, e.g., Ayers at column 4, lines 1-4 and column 6, lines 53-58.

Independent claims 1, 8, 15, 16, and 18 included limitations corresponding to a user trace data command in which execution of the user trace data command results in writing user trace data to a user trace data register. Applicant respectfully submits that Ayers does not teach or suggest these elements. The instrumentation code of Ayers is not a user trace data command and its execution does not result in writing user trace data. While execution of the instrumentation code of Ayers writes block identifiers to memory, the block identifiers are not trace data. Additional processing is required to convert the written block identifiers into a trace record. See, e.g., Ayers at column 4, lines 1-4;

column 6, lines 53-58. Moreover, the instrumentation code of Ayers does not permit a user to define trace data of interest.

Independent claims 1, 8, 15, 16, and 18 also include the limitation that the trace record includes at least a part of the user trace data written in the user trace data register in response to execution of the user trace data command. Ayers does not satisfy this element. As previously described, the instrumentation code of Ayers generates a block instruction sequence. However, the block instruction sequence is not traced out. Additional post-processing is performed by post-processors 217 and 317 on the block sequence to generate trace records. This includes the post processor writing out the sequence record of execution of the blocks and replacing each entry in a block with the set of program counters for the block in order to produce a trace record of program counter values. See, e.g., Ayers at column 4, lines 1-4; column 6, lines 53-58 and column 6, lines 53-63 and column 7, lines 9-14.

Additionally, independent claims 1, 8, 15, 16, and 18 include limitations corresponding to generating a trace record in response to detecting a write to a user trace data register of user trace data. Ayers does not teach or suggest the limitation of generating a trace record in response to detecting a write of user trace data. In contrast, Ayers teaches that a trace record is generated in response to a post-processor 217, 317 detecting a crash or user-defined event. In response to the post processor 217, 317 detecting an event, the post processor proceeds with writing out the sequence record of execution of the blocks and replacing each entry in a block with the set of program counters for the block in order to produce a trace record of program counter values. See, e.g., Ayers at column 4, lines 1-4; column 6, lines 53-58 and column 6, lines 53-63 and column 7, lines 9-14.

The dependent claims are allowable for at least the same reasons and include additional limitations. In view of the foregoing amendments and remarks, it is respectfully submitted that the application is now in condition for allowance. The Examiner is invited to contact the undersigned if there are any residual issues that can be resolved through a telephone call.

The Commissioner is hereby authorized to charge any appropriate fees to Deposit Account No. 03-3117.

Dated: April 14, 2004

Respectfully submitted,
COOLEY GODWARD LLP

By: [illegible]

Edward A. Van Gieson
Reg. No. 44,386

Cooley Godward LLP
ATTN: Patent Group
Five Palo Alto Square
3000 El Camino Real
Palo Alto, CA 94306-2155
Tel: (650) 843-5000
Fax: (650) 857-0663